A Comparison of Peer-to-Peer Query Response Modes

Wolfgang Hoschek
CERN IT Division
European Organization for Nuclear Research
1211 Geneva 23, Switzerland
`wolfgang.hoschek@cern.ch`

ABSTRACT

In a large distributed system spanning many administrative domains such as a Grid, it is desirable to maintain and query dynamic and timely information about active participants such as services, resources and user communities. However, in such a database system, the set of information tuples in the universe is partitioned over one or more distributed nodes, for reasons including autonomy, scalability, availability, performance and security. This suggests the use of Peer-to-Peer (P2P) query technology. A variety of query response modes can be used to return matching query results from P2P nodes to an originator. Although from the functional perspective all response modes are equivalent, no mode is optimal under all circumstances. Which query response modes allow to express suitable trade-offs for a wide range of P2P applications?

In this paper, we answer this question by systematically describing and characterizing four query response modes for the *Unified Peer-to-Peer Database Framework (UPDF)* proposed in our prior studies, namely *Routed Response*, *Direct Response*, *Routed Metadata Response*, and *Direct Metadata Response*. The response models are compared with respect to distribution and location transparency, efficiency of query support, economics, number of TCP connections at originator and agent, latency, caching and trust delegation to unknown parties. We discuss to what extent a given P2P network must mandate the use of any particular response mode throughout the system. As a result, we propose that response modes can be mixed by *switches* and *shifts*, in arbitrary permutations.

KEY WORDS

Peer-to-Peer Networks, Messaging, Service Discovery

1 Introduction

In a large distributed system spanning administrative domains such as a Grid [1], it is desirable to maintain and query dynamic and timely information about active participants such as services, resources and user communities. Other examples are a (worldwide) service discovery infrastructure for a multi-national organization, a Peer-to-Peer (P2P) file sharing system, the Domain Name System (DNS), the email infrastructure, a monitoring infrastructure for a large-scale cluster of clusters, or an instant messaging and news service. For example, the European DataGrid (EDG) [2, 3] is a software infrastructure that ties together a massive set of globally distributed organizations and computing resources for data-intensive physics applications, including thousands of network services, tens of thousands of CPUs, WAN Gigabit networking as well as Petabytes of disk and tape storage [4].

An enabling step towards increased Internet and Grid software execution flexibility is the *web services* vision [2, 5, 6] of distributed computing where programs are no longer configured with static information. Rather, the promise is that programs are made more flexible and powerful by querying Internet databases (registries) at runtime in order to discover information and network attached third-party building blocks. Services can advertise themselves and related metadata via such databases, enabling the assembly of distributed higher-level components.

In support of this vision we have introduced the *Web Service Discovery Architecture (WSDA)* [7] and given motivation and justification [8] for the assertion that realistic ubiquitous service and resource discovery requires a rich general-purpose query language such as XQuery [9] or SQL [10]. Based on WSDA, we introduced the *hyper registry* [11], which is a centralized database (node) for discovery of dynamic distributed content.

However, in an Internet discovery database system, the set of information tuples in the universe is partitioned over one or more distributed nodes (peers), for reasons including autonomy, scalability, availability, performance and security. It is not obvious how to enable powerful discovery query support and collective collaborative functionality that operate on the distributed system as a whole, rather than on a given part of it. Further, it is not obvious how to allow for search results that are fresh, allowing time-sensitive dynamic content.

It appears that a Peer-to-Peer (P2P) database network may be well suited to support dynamic distributed database search, for example for service discovery. The overall P2P idea is as follows. Rather than have a centralized database, a distributed framework is used where there exist one or more autonomous database nodes, each maintaining its own data. Queries are no longer posed to a central database; instead, they are recursively propagated over the network to some or all database nodes, and results are collected and send back to the client.

Consequently, we devised the WSDA based *Unified Peer-to-Peer Database Framework (UPDF)* [2] and its associated *Peer Database Protocol (PDP)* [12], which are unified in the sense that they allow to express specific applications for a wide range of data types (typed or untyped XML, any MIME type [13]), node topologies (e.g. ring, tree, graph), query languages (e.g. XQuery, SQL), neighbor selection policies (in the form of an XQuery), pipelining characteristics, timeout and other scope options.

A *link topology* describes the link structure among nodes. For example, in a worldwide service discovery system, a link topology can tie together a distributed set of administrative domains, each hosting a registry node holding descriptions of services local to the domain. Several link topology models covering the spectrum from centralized models to fine-grained fully distributed models can be envisaged, among them single node, star, ring, tree, semi hierarchical as well as graph models. Figure 1 depicts some example topologies.

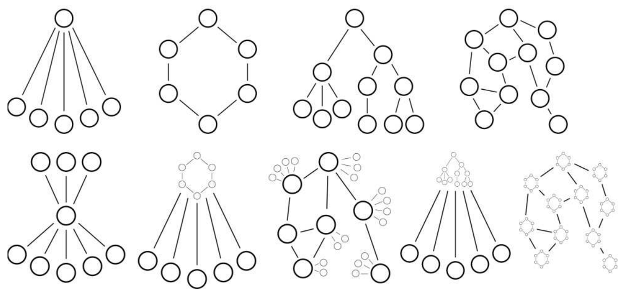

Figure 1. Example Link Topologies [14].

In the UPDF framework, an *originator* sends a query to an *agent* node, which evaluates it, and forwards it to select *neighbor nodes*. For reliable loop detection in query routes, a query has an identifier and a certain life time. To each query, an originator attaches an *abort timeout*, a *loop timeout* and a different *transaction identifier*, which is a universally unique identifier (UUID). A node maintains a state table of transaction identifiers and returns an error when a query is received that has already been seen and has not yet timed out.

A variety of query response modes can be used to return matching query results from P2P nodes to an originator. Although from the functional perspective all response modes are equivalent, no mode is optimal under all circumstances. The key problem then is:

- *Which query response modes allow to express suitable trade-offs for a wide range of P2P applications?*

In this paper, we answer the above question by systematically describing and characterizing four query response modes for the UPDF framework. Under *Routed Response*, results are fanned back into the originator along the paths on which the query flowed outwards. Each (passive) node returns to its (active) client not only its own local results but also all remote results it receives from neighbors. Under *Direct Response*, results are not returned by routing through intermediary nodes. Each (active) node that has local results directly invites the (passive) agent to retrieve results, which the agent then combines and hands back to the originator. Interaction consists of two phases under *Routed Metadata Response and Direct Metadata Response*. In the first phase, routed responses or direct responses are used. However, nodes return only small metadata results. In the second phase, the originator selects which data results are relevant. The originator directly connects to the relevant data sources and asks for data results.

This paper is organized as follows. Section 2 describes in detail the four response modes. Section 3 compares the properties of the various response models with respect to distribution and location transparency, efficiency of query support, economics, number of TCP connections at originator and agent, latency, caching and trust delegation to unknown parties. Section 4 discusses to what extent a given P2P network must mandate the use of any particular response mode throughout the system. We propose that response modes can be mixed by *switches* and *shifts*, in arbitrary permutations. Section 5 compares our work with existing research results. Finally, Section 6 concludes this paper.

2 Response Modes

We propose to distinguish four techniques to return matching query results to an originator: *Routed Response*, *Direct Response*, *Routed Metadata Response*, and *Direct Metadata Response*, as depicted in Figure 2. Let us examine the main implications with a Gnutella use case. A typical Gnutella query such as *"Like a virgin"* is matched by some hundreds of files, most of them referring to replicas of the very same music file. Not all matching files are identical because there exist multiple related songs (e.g. remixes, live recordings) and multiple versions of a song (e.g. with different sampling rates). A music file has a size of at least several megabytes. Many thousands of concurrent users submit queries to the Gnutella network. A large fraction of users lives on slow and unreliable dialup connections.

- **Routed Response.** (Figure 2-a). Results are propagated back into the originator along the paths on which the query flowed outwards. Each (passive) node returns to its (active) client not only its own local results but also all remote results it receives from neighbors. The response protocol is tightly coupled to the query protocol. Routing messages through a logical overlay network of P2P nodes is much less efficient than routing through a physical network of IP routers [15]. Routing back even a single Gnutella file (let alone all results) for each query through multiple nodes would consume large amounts of overall system bandwidth, most likely grinding Gnutella to a screeching halt.

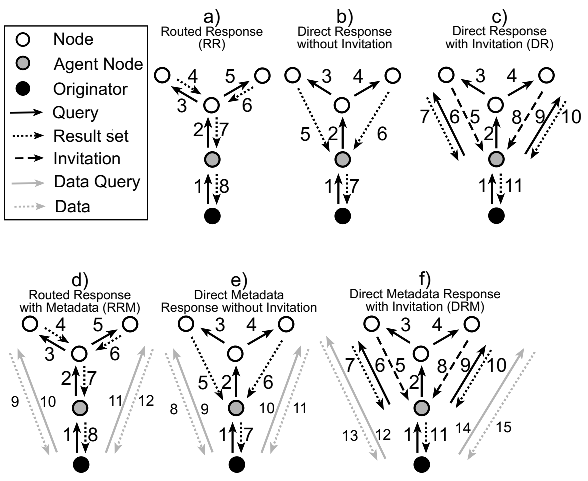
Node
Agent Node
Originator
Query
Result set
Invitation
Data Query
Data
a) Routed Response (RR)
b) Direct Response without Invitation
c) Direct Response with Invitation (DR)
d) Routed Response with Metadata (RRM)
e) Direct Metadata Response without Invitation
f) Direct Metadata Response with Invitation (DRM)

Figure 2. Peer-to-Peer Response Modes.

As the P2P network grows, it is fragmented because nodes with low bandwidth connections cannot keep up with traffic [16]. Consequently, routed responses are not well suited for file sharing systems such as Gnutella. In general, *overall economics* dictate that routed responses are not well suited for systems that return many and/or large results.

- **Direct Response With and Without Invitation.** To better understand the underlying idea, we first introduce the simpler variant, which is Direct Response Without Invitation (Figure 2-b). Results are not returned by routing back through intermediary nodes. Each (active) node that has local results sends them directly to the (passive) agent, which combines and hands them back to the originator. Response traffic does not travel through the P2P system. It is offloaded via individual point-to-point data transfers on the edges of the network. The response push protocol can be separated from the query protocol. For example, HTTP, FTP or other protocols may be used for response push. Let us examine the main implications with a use case.

  As already mentioned, a typical Gnutella query such as *"Like a virgin"* is matched by some hundreds of files, most of them referring to replicas of the very same music file. For Gnutella users it would be sufficient to receive just a small subset of matching files. Sending back *all* such files would unnecessarily consume large amounts of direct bandwidth, most likely restricting Gnutella to users with excessive cheap bandwidth at their disposal. Note however, that the overall Gnutella system would be only marginally affected by a single user downloading, say, a million music files, because the largest fraction of traffic does not travel through the P2P system itself.

  In general, *individual economics* dictate that direct responses without invitation are not well suited for systems that return many equal and/or large results, while a small subset would be sufficient. A variant based on invitation (Figure 2-c) softens the problem by inverting control flow. Nodes with matching files do not blindly push files to the agent. Instead they invite the agent to initiate downloads. The agent can then act as it sees fit. For example, it can filter and select a subset of data sources and files and reject the rest of the invitations. Due to its inferiority, the variant without invitation is not considered any further. In the remainder of this thesis, we use the term Direct Response as a synonym for Direct Response With Invitation.

- **Routed Metadata Response and Direct Metadata Response.** Here interaction consists of two phases. In the first phase, routed responses (Figure 2-d) or direct responses (Figure 2-e,f)) are used. However, nodes do not return data results in response to queries, but only small metadata results. The metadata contains just enough information to enable the originator to retrieve the data results and possibly to apply filters before retrieval. In the second phase, the originator selects, based on the metadata, which data results are relevant. The (active) originator directly connects to the relevant (passive) data sources and asks for data results. Again, the largest fraction of response traffic does not travel through the P2P system. It is offloaded via individual point-to-point data transfers on the edges of the network. The retrieval protocol can be separated from the query protocol. For example, HTTP, FTP or other protocols may be used for retrieval.

  The routed metadata response approach is used by file sharing systems such as Gnutella. A Gnutella query does not return files; it just returns an annotated set of HTTP URLs. The originator connects to a subset of these URLs to download files as it sees fit. Another example is a service discovery system where the first phase returns a set of service links instead of full service descriptions. In the second phase, the originator connects to a subset of these service links to download service descriptions as it sees fit. Another example is a *referral* system where the first phase uses routed metadata response to return the service links of the set of nodes having local matching results ("*Go ask these nodes for the answer*"). In the second phase, the originator or agent connects directly to a subset of these nodes to query and retrieve result sets as it sees fit. This variant avoids the "invitation storm" possible under Direct Response. Referrals are also known as *redirections*. A metadata response mode with a radius scope of zero can be used to implement the referral behavior of the Domain Name System (DNS). For details, see Section 5.

3 Response Mode Properties

Let us compare the properties of the various response models. The following abbreviations are used. RR ... Routed Response, RRM ... Routed Response with metadata, RRX ... Routed Response with and without metadata, DR ... Direct Response, DRX ... Direct Response with and without metadata.

- **Distribution and Location Transparency.** In the response models without metadata, the originator is unaware that (and how) tuples are partitioned among nodes. In other words, these models are transparent with respect to distribution and location. Metadata responses require an originator to contact individual data providers to download full results, and hence are not transparent.

- **(Efficient) Query Support.** All models can answer any query. Both *simple* and *medium* queries can be answered efficiently by RRX and DRX, whereas a *complex* query cannot be answered efficiently [2]. Transmission of duplicate results unnecessarily wastes bandwidth. RRX can eliminate duplicates already along the query path, whereas DRX can only do so in the final stage, at the agent. Similarly, maximum result set size limiting is more efficient under RRX because superfluous results can already be discarded along the query path.

- **Economics.** RR results travel multiple hops rather than just a single hop. This leads to poor *overall economics*. The effect is more pronounced for large results, as is the case for music files. RR can also lead to unfortunate *individual economics*. A user that induces few or undemanding queries consumes few system resources. However, if many heavy results for queries from other parties are routed back via such a user's node, it can end up in a situation where it pays for large amounts of bandwidth and gives it away for free to anonymous third parties. For a given user, the costs may drastically outweigh the gains. One could perhaps devise appropriate authorization, quality of service and flow control policies. The unsatisfying economic situation is similar to the one of physical IP routers on the Internet, which also forward traffic from and to third parties. In any case, there remains the fact that results travel multiple hops rather than just one.

  In principle, RRM has the same poor economic properties as RR. However, if metadata is very small in size (e.g. as in Gnutella), then the incurred processing and transmission cost may be acceptable. For example, Gnutella nodes just route back an annotated set of HTTP URLs as metadata. Under DRX, result traffic does not travel through the P2P system. Retrieving results is a deal between just two parties, the provider and the consumer. Consequently, individual economics are controllable and predictable. A user is not charged much for other peoples workloads, unless he explicitly volunteers.

- **Number of TCP Connections at Originator.** Under RR and DR, just one (or no) TCP connection is required at the originator, whereas metadata modes require a connection per (selected) data provider. The more data sources are selected, the more heavyweight data retrieval becomes. Metadata modes can encounter serious latency limitations due to the very expensive nature of secure (and even insecure) TCP connection setup. Hence, the approach does not scale well. However, for many use cases this may not be a problem because a client always selects only a small number of data providers (e.g. 10).

- **Number of TCP Connections at Agent.** Usually a node has few neighbors (five to hundreds). Under RRX, one TCP connection per neighbor is required at an agent. Under DRX, additionally a connection per data provider is required. Again, the more data providers exist, the more heavyweight data retrieval becomes. DRX can encounter serious latency limitations due to the very expensive nature of secure (and even insecure) TCP connection setup. For example, a query that finds the total number of services in the domain `cern.ch` should use RRX. Under DRX, it may generate responses from every single node in that domain. Consequently, an agent can face an invitation storm resembling a denial of service attack. On the other hand, the potential to exploit parallelism is large. All data providers can be handled independently in parallel.

- **Latency.** If a query is of a type that cannot support pipelining, the latency for the first result to arrive at the originator is always poor. For a pipelined query, the latency for the first result to arrive is small under DRX, because a response travels a single hop only. Under RRX, a response travels multiple hops, and latency increases accordingly. However, the cost of TCP

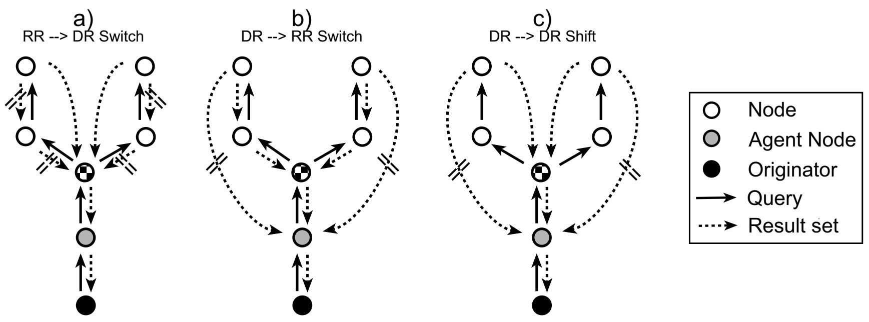

Figure 3. Response Mode Switches and Shifts.

connection setup at originator and/or agent can invert the situation. Under RR, the cost of TCP connection setup to nodes is paid only once (at node publication time), because connections can typically be kept alive until node deregistration. This is not the case under the other response modes.

- **Caching.** Caching is a technique that trades content freshness for response time. RRX can potentially support caching of content from other nodes at intermediate nodes because response flow naturally concentrates and integrates results from many nodes. DRX nodes return results directly and independently, and hence cannot efficiently support caching.

- **Trust Delegation to Unknown Parties.** Query and result traffic are subject to security attacks. It is not sufficient to establish a secure mutually authenticated channel between any two nodes because malicious nodes can divert routes or modify queries and results. Since a query is almost always routed through multiple hops, many of which are unknown to the agent, we believe that indirect delegation of trust to unknown parties cannot practically be avoided. Security sensitive applications should choose DRX because at least the retrieval of results occurs in a predictable manner between just two parties that can engage in secure mutual authentication and authorization. RRM merely delegates trust on metadata results, but not on full results.

4 Response Mode Switches and Shifts

Although from the functional perspective all response modes are equivalent, clearly no mode is optimal under all circumstances. The question arises as to what extent a given P2P network must mandate the use of any particular response mode throughout the system. Observe that nodes are autonomous and defined by their interface only. A node does not "see" what kind of response mode (or technology in general) its neighbors use in answering a query. As long as query semantics are preserved, the node does not care. Consequently, we propose that response modes can be mixed by *switches* and *shifts*, in arbitrary permutations, as depicted in Figure 3.

- **Routed Response ⇒ Direct Response switch.** (Figure 3-a). Starting from the agent, Routed Response is used initially. The central node ("football") receives a query from the agent. For some reason, it decides to answer the query using Direct Response. The response flow that would have been taken under Routed Response is shown crossed out.

- **Direct Response ⇒ Routed Response switch.** (Figure 3-b). Initially, Direct Response is used. However, the "football" decides to answer the query using Routed Response.

- **Direct Response ⇒ Direct Response shift.** (Figure 3-c). Initially, Direct Response is used. The football decides to continue using Direct Response but shift the target of responses. To its own neighbors the football declares itself as (a fake) agent. The responses that would have flowed into the real agent now flow back into the football, and then from the football to the real agent. Note again that this does not break semantics because the football behaves as if the results would have been obtained from its own local database. The real agent receives the same results, but solely from the football.

- **Routed Response ⇒ Routed Response shift.** At each hop, the response target is shifted to be the current node. Interestingly, this kind of shift is at the very heart of the definition of routed response. The classification introduced here shows that this is not the only possible approach.

A node may choose its response mode based on a local and autonomous assessment of the advantages and disadvantages involved. However, because of its context knowledge, often the client (e.g. originator) is in the best position to judge what kind of response mode would be most suitable. Therefore, it is useful to allow specifying as part of the query a hint that indicates the preferred response mode (`routed` or `direct`).

5 Related Work

DNS. Distributed databases with a hierarchical name space such as the Domain Name System (DNS) [17] can efficiently answer queries of the form *"Find an object by its full name"*. These systems arrange the link topology, according to the hierarchical name space, as a tree topology. A query searching for the IP address of a domain name traverses the tree on the shortest path from originator to the node containing the domain name - first up, then down. At each node, a *name resolution* policy selects the neighbor "closer" to the name than the current node, according to name space metadata. In DNS, queries are not forwarded (routed) through the topology. Instead, a node returns a *referral* message that redirects an originator to the next closer node. The originator explicitly queries the next node, is referred to yet another closer node, and so on. The DNS referral behavior can be implemented within our framework by using a radius scope of zero. The same holds for the LDAP referral behavior (see below).

X.500, LDAP and MDS. The hierarchical distributed X.500 directory [18] works similarly to the DNS. It also supports referrals, but in addition can forward queries through the topology, using routed response (*chaining* in X.500 terminology). LDAP [19] is a simplified subset of X.500. Like DNS, it supports referrals but not query forwarding. The Metacomputing Directory Service (MDS) [20] inherits all properties of LDAP. MDS additionally implements a simple form of query forwarding with routed response that allows for multi-level hierarchies but not for arbitrary topologies. Here neighbor selection forwards the query to LDAP servers overlapping with the name space.

6 Conclusions

A variety of query response modes can be used to return matching query results from P2P nodes to an originator. Although from the functional perspective all response modes are equivalent, no mode is optimal under all circumstances. Which query response modes allow to express suitable trade-offs for a wide range of P2P applications?

In this paper, we answer this question by systematically describing and characterizing four query response modes for the *Unified Peer-to-Peer Database Framework (UPDF)* proposed in our prior studies, namely *Routed Response*, *Direct Response*, *Routed Metadata Response*, and *Direct Metadata Response*. The response models are compared with respect to distribution and location transparency, efficiency of query support, economics, number of TCP connections at originator and agent, latency, caching and trust delegation to unknown parties. We discuss to what extent a given P2P network must mandate the use of any particular response mode throughout the system. As a result, we propose that response modes can be mixed by *switches* and *shifts*, in arbitrary permutations.

References

[1] Ian Foster, Carl Kesselman, and Steve Tuecke. The Anatomy of the Grid: Enabling Scalable Virtual Organizations. *Int'l. Journal of Supercomputer Applications*, 15(3), 2001.

[2] Wolfgang Hoschek. *A Unified Peer-to-Peer Database Framework for XQueries over Dynamic Distributed Content and its Application for Scalable Service Discovery*. PhD Thesis, Technical University of Vienna, March 2002.

[3] Wolfgang Hoschek, Javier Jaen-Martinez, Asad Samar, Heinz Stockinger, and Kurt Stockinger. Data Management in an International Data Grid Project. In *1st IEEE/ACM Int'l. Workshop on Grid Computing (Grid'2000)*, Bangalore, India, December 2000.

[4] Large Hadron Collider Committee. Report of the LHC Computing Review. Technical report, CERN/LHCC/2001-004, April 2001. http://cern.ch/lhc-computing-review-public/Public/Report_final.PDF.

[5] Ian Foster, Carl Kesselman, Jeffrey Nick, and Steve Tuecke. The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration, January 2002. http://www.globus.org.

[6] P. Cauldwell, R. Chawla, Vivek Chopra, Gary Damschen, Chris Dix, Tony Hong, Francis Norton, Uche Ogbuji, Glenn Olander, Mark A. Richman, Kristy Saunders, and Zoran Zaev. *Professional XML Web Services*. Wrox Press, 2001.

[7] Wolfgang Hoschek. The Web Service Discovery Architecture. In *Proc. of the Int'l. IEEE/ACM Supercomputing Conference (SC 2002)*, Baltimore, USA, November 2002. IEEE Computer Society Press.

[8] Wolfgang Hoschek. A Data Model and Query Language for Service Discovery. Technical report, DataGrid-02-TED-0409, April 2002.

[9] World Wide Web Consortium. XQuery 1.0: An XML Query Language. *W3C Working Draft*, December 2001.

[10] International Organization for Standardization (ISO). Information Technology-Database Language SQL. *Standard No. ISO/IEC 9075:1999*, 1999.

[11] Wolfgang Hoschek. A Database for Dynamic Distributed Content and its Application for Service and Resource Discovery. In *Int'l. IEEE Symposium on Parallel and Distributed Computing (ISPDC 2002)*, Iasi, Romania, July 2002.

[12] Wolfgang Hoschek. A Unified Peer-to-Peer Database Protocol. Technical report, DataGrid-02-TED-0407, April 2002.

[13] N. Freed and N. Borenstein. Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies. *IETF RFC 2045*, November 1996.

[14] Nelson Minar. Peer-to-Peer is Not Always Decentralized. In *The O'Reilly Peer-to-Peer and Web Services Conference*, Washington, D.C., November 2001.

[15] Matei Ripeanu. Peer-to-Peer Architecture Case Study: Gnutella Network. In *Int'l. Conf. on Peer-to-Peer Computing (P2P2001)*, Linkoping, Sweden, August 2001.

[16] Clip2Report. Gnutella: To the Bandwidth Barrier and Beyond. http://www.clip2.com/gnutella.html.

[17] P. Mockapetris. Domain Names - Implementation and Specification. *IETF RFC 1035*, November 1987.

[18] International Telecommunications Union. Recommendation X.500, Information technology – Open System Interconnection – The directory: Overview of concepts, models, and services. *ITU-T*, November 1995.

[19] W. Yeong, T. Howes, and S. Kille. Lightweight Directory Access Protocol. *IETF RFC 1777*, March 1995.

[20] Karl Czajkowski, Steven Fitzgerald, Ian Foster, and Carl Kesselman. Grid Information Services for Distributed Resource Sharing. In *Tenth IEEE Int'l. Symposium on High-Performance Distributed Computing (HPDC-10)*, San Francisco, California, August 2001.